Sohag J. Sci. 2, No. 3, 27-40 (2017)

Sohag Journal of Science
An International Journal

http://dx.doi.org/10.18576/sjs/020301

Map Reduce Frequent Sub graphs Mining on Cloud System

Marghny H. Mohamed[1,*], *Hosam E. Refaat*[2] *and Hanan H. Amin*[3].

[1]Dept. of Computer Science, Faculty of Computers and Information, *Assiut* University, Egypt.
[2]Dept. of Information System, Faculty of Computers and Informatics, Suez Canal University, Egypt.
[3]Dept. of Math, Faculty of Science, Sohag University, Egypt

Received: 14 Jul. 2017, Revised: 22 Aug.2017, Accepted: 28 Aug.2017.
Published online: 1 Sep. 2017.

Abstract: Analyzing frequent subgraph mining (FSM) is considered as the most important challenge to graph mining domain. Many algorithms have been proposed for this problem. The plurality of these algorithms assumes that the graph data can be handled in computer memory. Actually, FSM is a primal operation in many applications such as Social Networks or chemical components, which contains a huge number of edges and vertices. The previous algorithms give insufficient solutions for the massive data. Accordingly, MapReduce paradigm introduces a distributed solution to massive data computation. Hence, the proposed algorithm in this paper, which is called MRFSG, uses an iterative MapReduce-based framework. Moreover, MRFSG is balanced the load among the system workers and reduces dependency between the workers. Our experiments evaluate the performance of MRFSG using various of datasets. The results of experiment demonstrate that the proposed algorithm can scale well and efficiently process large graph datasets on the cloud system.
Keywords: Graph mining, Frequent subgraph mining, Parallel system, FSG Algorithm.

1 Introduction

Finding interesting patterns in large data has a various range of applications. Also, the massive growth of the large data complexity creates a big challenge to the mining algorithms. Graph mining has attracted much attention due to the explosive growth in the bringing forth of graph databases. Graph mining is a well-known subject in data mining and machine learning. There are numerous implementations of graph mining, such as molecular substructure exploration [1], web link analysis [2], outlier detection [3], chemical molecules [4], and social networks .

One way of finding interesting item sets is by simply looking at its frequency. Frequent subgraph mining (FSM) is an essential part of graph mining. FSM aims to discover all frequent subgraphs in a given graph dataset. A subgraph is called frequent if its occurrence is no less than a predefined threshold [5] [9]. Through the last decades, frequent subgraph mining has been a significant theme in graph mining. Many studies were dedicated to this area, resulting in tremendous progress, in frequent item set mining, sequential pattern mining and so forth .

Since databases expand rapidly in both of its dimensions, the core issue in a frequent graph mining algorithm is the ability to analyze very large databases [6]. Frequent subgraphs are a subsidiary way which aims at characterizing graph datasets, building structural indices, classifying and clustering graphs. The graph database is a type of database that consists of either a multi-graph or single-graph. This paper concentrates on frequent subgraph mining (FSM) especially on the parallel single-graph setting. In a single large graph, most existing work has a database of numerous little graphical records. This is useful in several real-life applications such as protein interactions, citation graphs, social networks [7] which are modeled as a single large graph [8]. There are numerous algorithms for finding frequent subgraphs mining in a single large graph [9], most of them, however, use sequential strategies which are not working probably in the case of large dataset [5], especially in terms of run-time performance, for such very large databases. These algorithms are computationally concentrating on the graph isomorphism and the subgraph isomorphism issue.

In this paper, a parallel algorithm is proposed to find frequent subgraphs for a single large graph using a MapReduce model in a cloud system. There are a lot of features of cloud strategy, the most fundamental ones are lower costs, re-provisioning of assets and remote accessibility. Cloud computing minimizes the costs by avoiding the capital expenditure of the

*Corresponding author e-mail: marghny@aun.edu.eg

© 2017 NSP
Natural Sciences Publishing Cor.

company in renting the physical infrastructure from a third party provider. Due to the flexible nature of cloud computing, we can quickly access more resources from cloud providers when we need to expand our business. The remote accessibility helps to access the cloud services from anywhere at any time. Also, the run-time in MapReduce framework helps divide the input data and scheduling the program's execution over an arrangement of machines.

It is well known that Gspan [11] and Gaston [12] are additionally proficient algorithms in substructure design mining and their exhibitions are little superior to Apriori. In addition, these algorithms are more difficult to perform than other approaches, as complex data structure needs more memory than a list or array of transactions. On the other hand, recursive construction of the FP-trees and complex data structure requires large space [13].Thus, adjustment of A prior to the parallel system on account of its good parallel and scale up properties is one of the most important considerations in our work. The advantages of Apriori method are: big data which can be utilized, the easiness to parallelize, the easiness to recover the failed work and the reduce dependency between workers.

The rest of this paper is divided as follows: Section 2 gives related work overview. Section 3 introduces some basic concepts such as frequent subgraphs, Apriori algorithm and MapReduce model. Sections 4 describes the proposed algorithm (MRFSG). Section 5 presents experimental results of the algorithm performance measurements. The last section discusses the conclusions and future work.

2 Related Work

There are many sequential algorithms which enumerate frequent sub graphs mining. Traditional methods don't give much attention to the VM capability or required memory size for each FSM task. AGM [10], FSG [14], GS pan [11], Gaston [12], and DMTL [15] are the most important techniques which are able to process just a small amount of data, it mine a reasonable amount of time and assume that mining task is fit in the main memory of a computer. These methods will fail if data grows substantially and some other methods such as DB-FSG [16] and OOFSG [17] are considered as a large database system. Since the sequential methods fail to give a desirable performance, the parallel techniques are used to save time and memory in big databases to get frequent patterns.

Cook et al. [18] proposed a framework, where they explored data parallelism and functional parallelism and they used three paradigms, namely functional partitioning of the search space, dynamic partitioning of the workload and static partitioning of the dataset across nodes of the system.

In 2004, Wang and Parthasarathy [19] built up a parallel calculation for their Motif Miner toolbox. Motif Miner searched for interesting substructures in noisy geometric graphs (or graphs embedded in a 3-dimensional space) targeted at large biochemical molecules such as proteins. However, their parallelization design cannot be directly applied to the more general graph mining problem [20].

In 2005, Buehrer et al. [21] designed a parallel approach for Gspan. They developed a parallel algorithm for graph mining on the shared memory architecture. They assessed three models, namely, global queues, hierarchical queues and distributed queues.

In 2006, Meinl et al. [22] examined the parallel relation of Mofa and Gspan. Fatta and Berthold [23] also discussed the distributed approach to frequent subgraph mining using partitioning of the database, distributed task queue with dynamic load balancing and the peer to peer communication correspondence system. Reinhardt and Karypis broadened this work by paralleling the VSiGram calculation [24].

In 2012, Abhik Ray and Lawrence B. Holder's [25] developed the work done by Cook et al .to enhance the proficiency of MPI SUBDUE by adjusting the evaluation stage. Their examinations demonstrated the change in speed-up while holding the features of serial SUBDUE.

Also, in 2012, KhadidjaBelbachir and HafidaBelbachir [26] proposed a sequential algorithm "Partition" with a parallel version. Their method was different from others sequential algorithms in that the database was scanned only twice to get the significant association rule. Their parallel system didn't require much communication between the nodes.

In 2013, Ning Li et al.[27] proposed a parallel Apriori method called "PApriori" based on MapReduce. They used the size up, speed up and scale up to evaluate the performances of PApriori. They mine association rules from large databases. Their experimental results show that the program is actually more efficient as the database size is increased.

Recently, Kessl et al., [28] used CUDA to mine graph-based substructure patterns on GPUs. In addition, some studies have tried to parallelize FSM algorithms [29]. It can be seen that applying parallelism to FSM is an emerging trend.

In 2016, X. Zhao et al.[30] presented distributed algorithm based on Pregel for a single graph. The computed methods on the master and vertex were designed for working together to achieve the goal. Moreover, in order to enhance the mining performance, they had two techniques to reduce message passing and number of super steps.

K. M. Padmapriya and M. Keerthana in 2016 [31], introduced a method in frequent subgraph mining algorithm called FSM-H which uses an iterative MapReduce based framework. Their paper focused on efficient implementation of FSM over single large graph on a Pregel like extensible computing platform. To the best of our knowledge, this is among the first attempts to address the problem at scale under a modern distributed programming framework.

© 2017 NSP
Natural Sciences Publishing Cor.

Most of the previous methods don't adopt any mechanism to avoid generating duplicate patterns (subgraphs isomorphism) and have load balancing between workers. They also have a complicated model to get frequent subgraph mining and consume more time to mining big database. In this paper, we try to solve these problems.

3 Basic Concepts

In this section, we present some basic concepts such as frequent sub graph mining, Apriori algorithm and Map Reduce model.

3.1 Frequent Sub Graph Mining (FSM)

In graph exchange based FSM, the input data is either an accumulation of medium size graph called transactions or single based graph FSM, as the name suggests, is one substantial graph. Let DB be a graph database, each graph

G = (V, E) of DB, is given as a combination of nodes V and edges E, |V |is the number of nodes of G and |E| is the number of edges of G (also called graph size). If two nodes u ϵ V and v ϵ V and {u, v} ϵ E then u and v are said to be adjacent nodes, [4]. The goal of FSM is to discover all frequent sub graphs in a given graph dataset. A subgraph is called frequent if its event is over a user indicated threshold ɵ.

In the traditional parallel system, the database is described by an array with two dimensions. You can parcel two-dimensional array horizontally allocating columns of the premier array to the various workers or vertically by appointing rows, divide the number of columns or rows by the number of works as shown in Fig. 1. But these horizontal or vertical partitions don't reduce redundancy. Moreover, when any workers fail, a long time is needed to recover, this is what concerns us in this research.

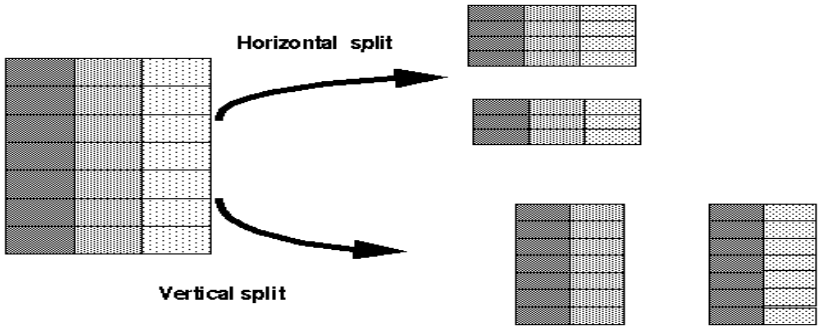


Figure 1. Horizontal and Vertical partitions

3.2 Apriori Algorithm

Apriori is one of the most classical algorithms presented by R.Agrawal and R.Srikant in 1994 for mining frequent items for Boolean association rule [32] [33]. It generates the k-candidate by combining two frequent (k − 1) item sets in the level-wise procedure. In this way, only the frequent item sets at a level are used to construct candidates at the next level [34]. The Apriori based approach utilizes the breadth first search BFS technique due to its level shrewd candidate generation. The pattern growth approach can utilize both BFS and depth-first search (DFS) [5]. Fig. 2 demonstrates Apriori and pattern growth strategies.

© 2017 NSP
Natural Sciences Publishing Cor.

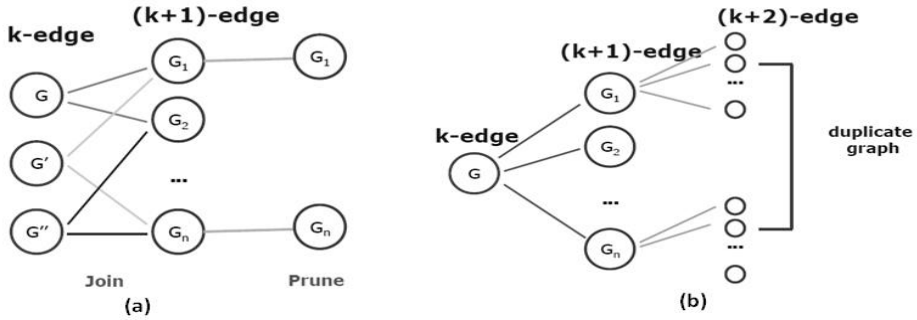


Figure 2. (a) Apriori- based vs. **(b)**pattern growth- based approach

This paper is designed along the lines of Apriori algorithm concentrated on FSG discovery algorithm. The FSG embraces edge-based candidate generation strategy and two graphs of size "k" are a unit to form resultant graphs of size "k+1" which ought to likewise be frequent. So, in each iteration, the size and the number of subgraph candidates is increased. Candidate pruning is additionally done if the produced candidate does not fulfill the minimum threshold. The FSG algorithm was introduced in [14] [35] as shown in Fig. 3.

```
//FSG Algorithm
Function FSG(D,min_sup)
//D-Graph database
// min_sup= minimum support
[
F^1→detect all frequent 1- subgraphs in D
F^2→detect all frequent 2- subgraphs in D
K→3
While F^(k-1)≠Null   do
C^k→fsg_gen(k-1)
 For each candidate g^k∈C^k  do
        g^k.count←0;
        For each graph g∈D  do
          If [candidate g^k∈g] then
              g^k.count→g^k.count+1
            End If
        End For
    End For
f^k←{g^k∈C^k: g^k.count≥min_sup};
  K←K+1
End While
Returnf^1, f^2, …,f^(k-2)
]// End function
```

Figure 3. FSG algorithm

whereCk is a set of candidates with k edge, gk is a k-(sub) graph, Fk is a set of frequent k-subgraphs and k-(sub)graph is a (sub) graph with k edges. The FSG method increments the size of a frequent dataset by inserting one edge at a time. In the FSG, frequent 1-edge and 2-egde graphs dataset are specified. Then, based on those two sets, it starts the main computational loop. During every iteration, it initially creates candidate subgraphs whose size is greater than the past frequent ones by one edge. Next, it tallies the frequency for each of these candidates and prunes subgraphs that don't fulfill the support requirement. Found frequent subgraphs fulfill the descending closure property of the support condition which

© 2017 NSP
Natural Sciences Publishing Cor.

enables us to successfully prune the lattice section of frequent subgraphs [35].

The Apriori algorithm (as well as FSG algorithm) is the simplest algorithm. However, it is costly in terms of time complexity and space, since it requires a repetitive scan of the database. With the speedy growth of the Internet and sensor network, data are increasing exponentially, which often causes non-negligence problems of memory overflow and huge delay in communication. Researchers propose many algorithms to overcome the above problems by using parallel frequent subgraphs mining algorithms to speed up the mining of the ever-increasing sized databases [36].

3.3 Map Reduce Model

MapReduce is a distributed processing model introduced by Google [27]. It is a parallel model for processing big data in a distributed massively parallel way using simple commodity system. It is a programming model and related application for handling and generating large datasets in an enormously parallel way. MapReduce has turned into a famous model for ad-van cements in cloud computing. It gives users a standard model to programming distributed algorithm, and it handles every one of the points of data distribution, load balancing, fault tolerance and replication. It gives a programming model a map function that procedures a key/value pair to produce a set of middle key/value sets, and a reduction function that merges every average value related with a similar transitional key, as seen in Fig.4.

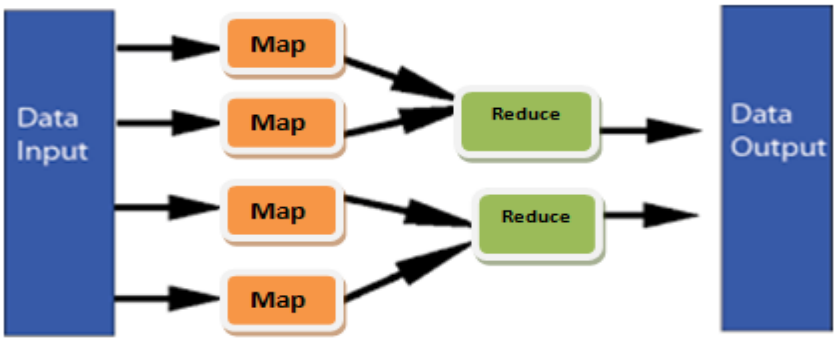


Figure 4. Map Reduce model

Mapping phase: is distributed across multiple machines by automatically partitioning the input data into a set of splits or shards. The input shards can be processed in parallel on different machines.

Reduce phase: After the Map phase is over, all the intermediate values for a given output key are combined together into a list and returns one or more final values for that same intermediate key. In short, this phase summarizes the complete dataset.

4 Proposed Method (MRFSG)

In this paper, we present a new algorithm, named MRFSG, for finding all frequent subgraph mining in a single large graph. The MRFSG algorithm is carried based on dividing the single large graph into a set of subgraphs and distributing it to parallel workers for two reasons: first, the large graphs take large amount of memory in sequential implementation, second, a parallel graph partitioning algorithms can take advantage of the extensively higher amount of memory available in parallel implementation to partition very large graphs.

© 2017 NSP
Natural Sciences Publishing Cor.

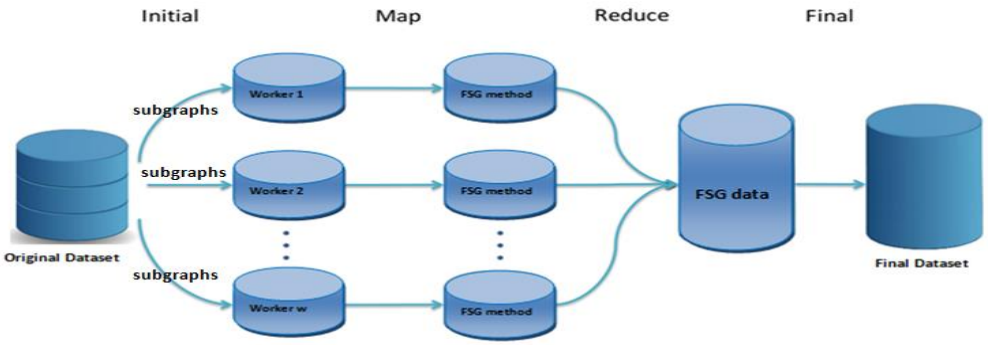


Figure 5. Map Reduce model

A large single graph is divided horizontally into sets of related sub graphs, which will be assigned to a number of the system workers *w*. This proposed method is an efficient algorithm for paralleling the FSG algorithm based on Map Reduce model with considering worker resources as shown in Fig.6. MRFSG finds all sub graphs from the original single large graph by finding the isomorphism between these sub graphs. If the graph contains thousands of nodes, the size of data that should be processed will be millions of bits. This will cause challenges in memory and processing power. The distribution of this job among parallel workers will provide the desired processing power, but the worker memory size will cause a challenge. Any parallel platform can be homogeneous or heterogeneous. In the case of the heterogeneous system, the distribution of the tasks must consider the resources of each worker. In the case of the homogeneous parallel system, which is the paper interests, the memory size is equivalent in each worker. Hence, the horizontal distribution of the subgraphs graph matrix can be written:

$$v = Ceil\left(\frac{n}{w}\right) \rightarrow (1)$$

where n is the number of sub graphs.
To simplify manipulating the tasks in each worker, the worker will take its job portion as a set of two tasks. In another word, the reserved memory size for each VM must be two times the subgraph size. Hence, equation (1) can be written as follows:

$$v = Ceil\left(\frac{n}{2w}\right) \rightarrow (2)$$

Equation 2 supposes that the job will be distributed over the system workers in two rounds. This strategy gives the parallel algorithm numerous advantages in load balancing and failure recovery. For

example, if we have a graph with size (20*20) contains 20 subgraphs and 4 workers we partition our work to each work to have 3 subgraphs approximately as illustrated in Fig.7.

We must note that the homogeneity of the system will not guarantee the identical performance of all VMs[37]. The reason for the randomness of the performance caused by resource allocates or that assigns the required resources on different physical machines on the cloud. Moreover, if two VMs are allocated in the same data center, this will not guarantee that both VMs have an identical performance due to the problem caused by other system factors like memory contention. This leads us to distributethe tasks in two rounds and monitor the system workers such that if any worker becomes idle, the master worker assigns a new task to it as shown in Fig. 8. This will increase resources usage by distributing the load between the system nodes.

© 2017 NSP
Natural Sciences Publishing Cor.

Figure 6. An example for graph with size 20*20 and their segments to 4 workers

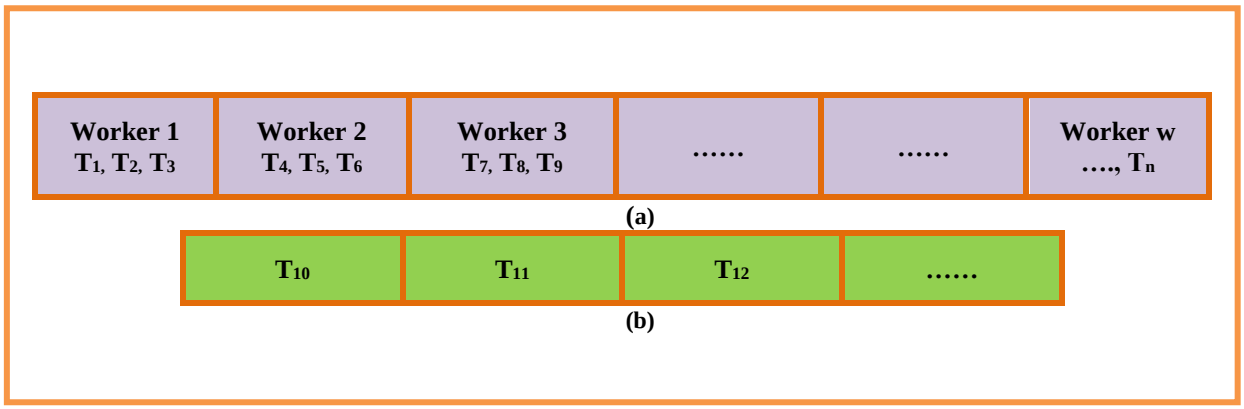


Figure 7. (a) Vectors of working with tasks and (b) Vectors of waiting tasks

The main advantages of the proposed model are the dependency reduction between system workers. This division reduces the defect recovery time in the case of worker failure. In another word, if the worker finishes its task and there are more tasks, it will start working on another task and will not be idle, otherwise, the waiting time of the slow worker is cut to half. Also, this division is more suitable for heterogeneous system resources. After finishing a set of tasks, its client is broadcasting the result, in which the system is called reduce, and then each worker collects its interest data.

As discussed previously, each worker in our example in Fig.7. will take three subgraphs (rows) the segments(tasks) from 1 to 4 are in process and from 5 to 7 are waiting. After each worker counts all items in its segment, it will broadcast the value of the frequent items and collect the value of its interesting items.

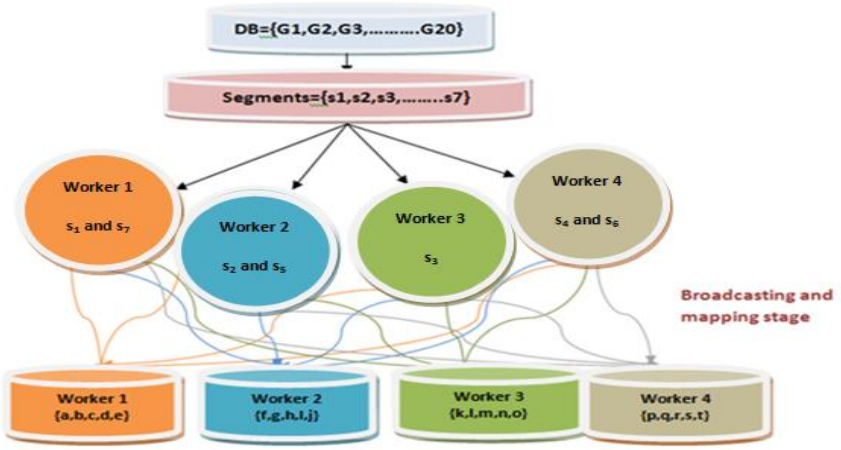
Suppose that, the candidate in the first stage is: c1={a,b,c, …., t}, each worker has one segment. The worker counts all candidates in its segment (i.e. Worker 1 counts the frequency c1in s1, worker 2 counts the frequency c1 in s2 etc.) After finishing its segment, the worker will not be idle; it will take another segment to compute frequency in it. The interesting table contains all candidates and these candidates are distributed to the worker equivalently as in Table 1.

From Fig.6, notice that, for example, worker 2 has two tasks (2,7). The worker will handle them as one segment; in other words, the worker will count all candidate items C1 in both segments as one segment and it will be interested in specific items as an interesting table (Ex: f,g,h,i,j). Hence, each worker will broadcast the frequency of all candidates except it's interesting one.

As it is mentioned, whatever the number of segments the worker has; the worker will deal with it as one segment. So, the worker will broadcast the frequent candidates of these segments without duplication in communication. Thus, each worker will accumulate the result of its assigned partitions until this stage is finished. Each stage is ended when there is no waiting tasks list. Hence, workers start to exchange the frequency data. Moreover, each worker will exclude broadcast of it's interesting information to reduce the communication. This process will be repeated in each stage with a new candidate until

© 2017 NSP
Natural Sciences Publishing Cor.

all frequent subgraphs are obtained as shown in Fig.8.



Interesting table for each worker	
Worker1	{a,b,c,d,e}
Worker2	{f,g,h,l,j}
Worker3	{k,l,m,n,o}
Worker4	{p,q,r,s,t }

Table 1. Intersting Table

Figure 8. Database segments and interested Table

Each worker is a fully functional copy of FSG with its own embedding lists and it starts to get first frequent, second frequent, etc until the (k-2) frequent subgraphs in the database. If any worker is idle, it takes a new task to complete the process.

MRFSG Algorithm

Input
G:graph database
S : minimum support
Output
Frequent all subgraphs

1. $x = subgraphs(G)$
2. T = isomorphism(x) //T is the tasks waiting list and $T \rightarrow \{t_1, t_2, \ldots\ldots, t_n\}$
3. $v = Ceil(\frac{|T|}{2w})$//Determine the worker quota for each round
4. // Get all frequent 1 and 2 subgraphs in G
5. **For** q =1:2 do
6. Broadcasting M{q} // where M{q} interesting Table to each worker.
7. **While**($T \neq \emptyset$)
8. **For** each worker w_i
9. **If** worker_Load(w_i)=0// worker is idle
10. Assign $(t_j, t_{j+1}, \ldots\ldots, t_{j+v}, w_i)$ // Assigning a set of subgraphs of size c to the idle worker w_i
11. **End If**
12. **End For**
13. **End While**
14. **If** reduce phase finished //all of the tasks T are finished and each worker broadcasting the frequent items values
15. $F^q = \cup^{\forall w_i} f_i$ // Combine all Frequent items from all workers
16. **Else**
17. Wait until all tasks are finished
18. Go to step 14
19. **End If**
20. **End For**
21. K→3
22. **While** $F^{k-1} \neq \emptyset$
23. Broadcasting M{k}

© 2017 NSP
Natural Sciences Publishing Cor.

24. $C^k \rightarrow$fsg_gen(k-1)
25. **For** each candidate $g^k \in C^k$
26. g^k.count←0;
27. **For** each graph g∈D
28. **If** [candidate $g^k \in g$] then
29. g^k.count→g^k.count+1
30. **End If**
31. **End For**
32. **End For**
33. **While**($T \neq \emptyset$))
34. **For** each worker $w_i$do
35. **If** worker_Load(w_i)=0
36. Assign($t_j, t_{j+1}, \ldots\ldots, t_{j+v}, w_i$)
37. **End If**
38. **End For**
39. **If** reduce phase finished
40. $F^k = \bigcup^{\forall w_i} f_i$,where $f_i \leftarrow \{g^k \in C^k: g^k.count \geq min\_sup\}$;
41. **Else**
42. Wait until all tasks are finished
43. Go to step 39
44. **End If**
45. **End While**
46. K←K+1
47. **End While**
48. Get all frequent subgraphs

Figure 9 . Proposed method

The initial step in this algorithm is to obtain subgraphs matrix from the single large graph and find the isomorphism between subgraphs. The result of this step is divided into tasks using equation 2 and tasks are written into a waiting list to be processed by the system workers (mapping phase). The system workers will apply FSG algorithm to count the frequency of all candidates subgraphs. Each worker is interested in a set of candidates, which concerning to collect its frequency from other workers. Each stage in finding the frequent item set ended when the waiting tasks are empty, which means starting exchange the worker data (reduce phase). In reduce phase, each worker broadcasts the frequency of all candidates except it's interesting items. Hence, each worker can determine the frequent items among its interesting items. Then the frequent items are determined by collecting the frequent items from the workers. The previous steps are repeated until finding all the frequent subgraphs.

5 Experimental Results

All experiments are conducted on ITI cloud computing system with a CPU of four workers, 4 GB of RAM, Intel (R) Xeon (R) CPU with 2.00 GHz 2.00 GHz (4 Processor) which runs Windows server 2008 R2 Enterprise. The researchers use different datasets for large single graph such as Karate , ca-GrQc dataset ,s-735 .HEP-TH . Also, the researchers generate random frequent single graph.

The algorithms were coded in Mat lab by using Mat lab 2012. The Characteristics of the experimental datasets are described in Table 2. The Table shows the size of graphs, the number of edges in the dataset, parallel FSG method, Belbachir method [26] and the proposed method with minimum support 0.3.

© 2017 NSP
Natural Sciences Publishing Cor.

Table 2. Characteristics of the experimental results

Num of nodes	Num of edges	Traditional Parallel FSG time (in a second)	Belbachir method time (in a second)	Proposed method time (in a second)	P1	P2
25	162	0.231907	0.230542	0.010942	<0.001**	<0.001**
50	596	10.645431	9.251497	7.138739	<0.001**	<0.002**
100	2444	80.450788	78.324781	42.295382	<0.001**	<0.001**
150	5592	1899.830128	1779.356874	1055.060501	<0.001**	<0.001**
200	10285	1925.088189	1794.214147	1135.537046	<0.001**	<0.001**
500	62174	3185.893210	3102.514287	2734.693428	<0.001**	<0.001**
1000	154654	9574.012456	9473.240145	8756.358716	<0.001**	<0.001**
2000	236521	18539.658471	18466.324185	17399.684512	<0.001**	<0.001**

P1: Comparison between the proposed method and the traditional parallel FSG
P2: Comparison between the proposed method and Belbachir method
**** Statistically significant difference (p<0.01)**

Table 2. shows that, in the comparison with the proposed method and traditional parallel FSG, the proposed method decreases the time with a statistically significant difference in time, Also in comparison with Belbachir method [32], our proposed method decreases the time with a statistically significant difference.

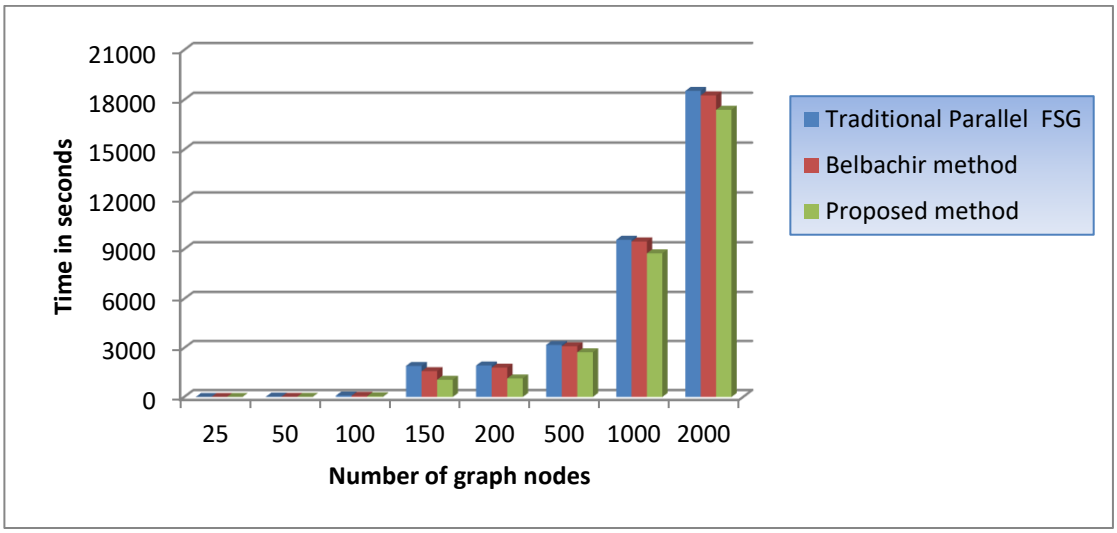


Figure 10 . Relation between traditional parallel FSG, Belbachir and proposed methods

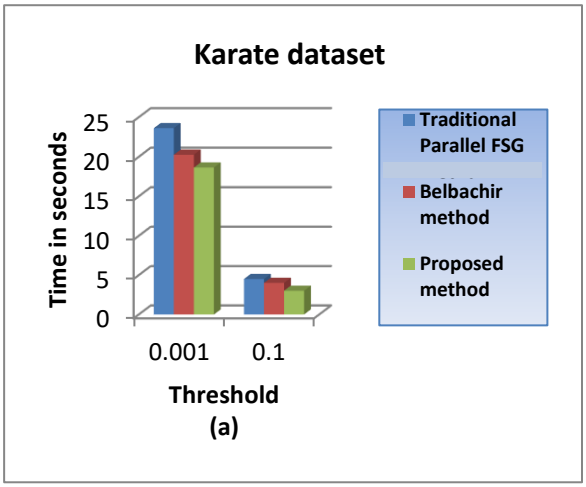
© 2017 NSP
Natural Sciences Publishing Cor.

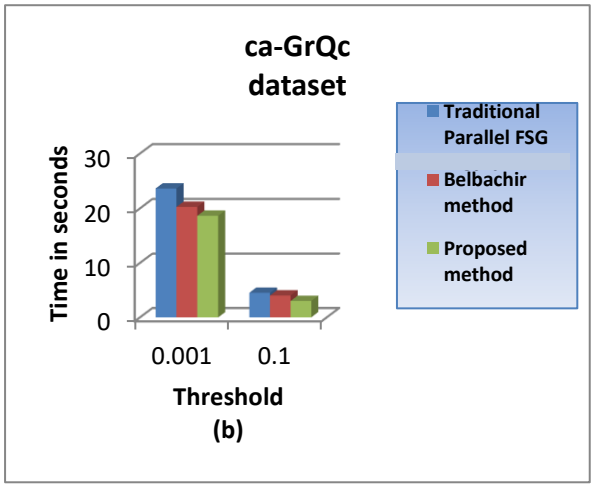
Table 3. Results with different datasets

Dataset	Num of nodes	Num of edges	Minimum support	Traditional Parallel FSG time (in a second)	Belbachir method time (in a second)	Proposed method time (in a second)	P1	P2
Karate	34	78	0.1 0.001	0.024142 0.157493	0.020145 0.1184210	0.016986 0.087493	0.022* 0.018*	0.043* 0.037*
ca-GrQc	5242	28980	0.1 0.001	4.509058 23.61356	4.015285 20.254311	3.015584 18.636608	0.027* 0.013*	0.042* 0.039*
As-735	7716	13233	0.01 0.001	10.564226 336.325547	9.502451 329.254174	7.389572 320.254589	0.005** 0.003**	0.009** 0.006**
ca-HepTh	9877	51971	0.01 0.001	19.254781 678.383727	15.2654871 663.251489	12.245178 653.608589	0.001** 0.001**	0.001** 0.001**

P1: Comparison between the proposed method and the traditional parallel FSG
P2: Comparison between the proposed method and Belbachir method
*** Statistically significant difference ($p<0.05$)**
**** Statistically significant difference ($p<0.01$)**

Table 3. shows that, in the comparison with the proposed method and the traditional parallel FSG, the proposed method decreases the time with statistically significant difference in time, Also in comparison with Belbachir method, our proposed method decreases the time with a statistically significant difference, as shown in Fig.11.





© 2017 NSP
Natural Sciences Publishing Cor.

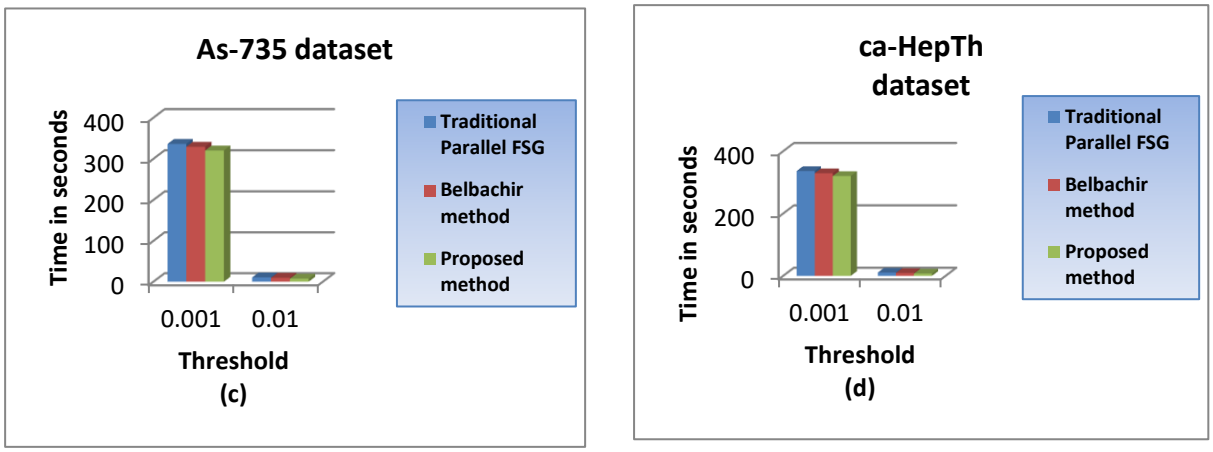


Figure 11. The relation between the traditional parallel FSG, Belbachire and the proposed methods for (a) Karate dataset, (b) ca-GrQc, dataset (c) As-735 dataset and (d) ca-Hepth dataset.

6 Conclusions

In this paper, we address the issue of the frequent subgraph mining process. We describe our proposed MRFSG approach for frequent subgraphs mining from a single large graph database. The proposed method uses MapReduce model in cloud system to parallel and builds balanced partitions of a graph database over a set of machines to save time and memory. By conducting experiments on a variety of datasets and randomly generated graphs, the researchers found that the proposed method decreased the time in a statistically significant difference from the time computed in the traditional parallel FSG and Belbachir methods. The performance and capability of our approach satisfying for large databases. In future work, the researchers will study the use of other techniques for cloud frequent subgraphs for a single large graph in the heterogeneous system.

References

[1] Nijssen, S. and Kok, J.: Frequent Graph Mining and its application to molecular databases. In: The IEEE Interna-tional Conference on Systems, Man and Cybernetics (SMC 2004), pp. 4571–4577, 2004.

[2] Punin, J.R., Krishnamoorthy, M.S., Zaki, M.J.: LOGML: Log markup language for web usage mining. In: Kohavi, R., Masand, B., Spiliopoulou, M., Srivastava, J. (eds.) WebKDD 2001.LNCS (LNAI) Springer, Heidelberg, vol. 2356, pp. 88–112, 2002.

[3] Eberle, W. and Holder, L.: "Anomaly detection in data represented as graphs." Intelligent Data Analysis 11, pp. 663–689, 2007.

[4] Dehaspe, L., Toivonen, H. and King, R.:" Finding Frequent Substructures in Chemical Compounds." In: KDD, pp. 30–36, 1998.

[5]Vo, B., Nguyen, D. and Nguyen, T., "A Parallel Algorithm for Frequent Subgraph Mining" Springer International Publishing Switzerland 2015 H.A. Le Thi et al. (eds.), Advanced Computational Methods for Knowledge Engineer-ing, page 163, Advances in Intelligent Systems and Computing 358, DOI: 10.1007/978-3-319-17996-4_15, 2015.

[6] Marghny, H. and Hosam, R.: A new parallel association rule mining algorithm on distributed shared memory sys-tem, Int. J. Business Intelligence and Data Mining, vol. 3, No.10, 2012.

[7] Nilothpal, T. and Mohammed, Z.: "A distributed approach for graph mining in massive networks", Data Min. Knowl. Disco. , vol. 5, No. 30, p.1024-1052,2016.

[8] Elseidy, M., Abdelhamid, E., Skiadopoulos, S. and Kalnis, P.:" GraMi: Frequent subgraph and pattern mining in a single large graph." PVLDB, 7, pp. 517–528, 2014.

© 2017 NSP
Natural Sciences Publishing Cor.

[9] Ray, A. and Holder, B.:" Efficiency improvements for parallel subgraph miners". In Florida Artificial Intelligence Research Society Conference, FLAIRS '12, 2012.

[10] Inokuchi, A., Washio, A. and Motoda, H.: "An apriori-based algorithm for mining frequent substructures from graph", In PKDD '00 Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pp. 13-23, 2000.

[11] Yan, X. and Han, J.:" gspan: Graph-based substructure pattern mining." In: The IEEE International Conference on Data Mining (ICDM 2002), pp. 721–724, 2002.

[12] Nijssen, S. and Kok, J.: Frequent graph mining and its application to molecular databases. In: The IEEE Interna-tional Conference on Systems, Man and Cybernetics (SMC 2004), pp. 4571–4577, 2004.

[13] Vimala, S., Kerana, D., Kaliyamurthie, P.: "A Study of Efficient Algorithms for Fast Recovery of Frequent Itemset Mining ", International Journal on Advanced Computer Engineering and Communication Technology, ISSN: 2319-2526, Vol. 5, Issue. 1, 2016.

[14]Kuramochi, M. and Karypis, G., " Frequent subgraph discovery", In Proc of ICDM, 2001.

[15] Chakravarthy, S. and Pradhan, S.:"Db-FSG: An SQL-based approach for frequent subgraph mining," in Proc. 19th Int. Conf. Database Expert Syst. Appl. , pp. 684-692, 2008.

[16] Parthasarathy, S. and Coatney, M.:" Efficient discovery of common substructures in macromolecules.", InPro-ceedings of the IEEE International Conference on Data Mining (ICDM), 2002.

[17] Meinl, T., Wörlein, M., Fischer, I. and Philippsen, M.:" Mining Molecular Datasets on Symmetric Multiprocessor Systems". In Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics 6: Taipei, Taiwan. IEEE Press, pp. 1269 – 1274, 2006.

[18] Cook, D.J., Holder L.B., Galal, G., and Maglothin, R.:" Approaches to parallel graph-based knowledge discovery.", Journal of Parallel Distrib. Comput. 61, Vol. 3, pp. 427-446, 2001.

[19] Wang, C. and Parthasarathy, S.:"Parallel algorithms for mining frequent structural motifs in scientific data." In Proceedings of the ACM, International Conference on Supercomputing (ICS), 2004

[20] Parthasarathy, S. and Coatney, M.:" Efficient discovery of common substructures in macromolecules.", InPro-ceedings of the IEEE International Conference on Data Mining (ICDM), 2002.

[21] Buehrer, G., Parthasarathy, S., Nguyen, A., Kim, D., Chen, Y. and Dubey, P.:"Parallel graph mining on shared memory architectures. " , Technical report, The Ohio State University, Columbus, OH, USA, 2005.

[22] Meinl, Thorsten, Fischer, Ingrid, Philippsen, Michael:"Parallel Mining for Frequent Fragments on a Shared-Memory Multiprocessor -Results and Java-Obstacles".In: Bauer, Mathias ; Kröner, Alexander ; Brandherm, Boris (Ed.): LWA 2005 - Beiträgezur GI-WorkshopwocheLernen, Wissensentdeckung, Adaptivität (Workshop der GI-Fachgruppe "MaschinellesLernen, Wissensentdeckung, Data Mining" (FGML), Saarbrücken, Germany, 2005-10-10 - 2005-10-12), pp. 196-201, 2005

[23] Di Fatta, G. and Berthold, M.R.:"Dynamic load balancing for the distributed mining of molecular structures.", IEEE Transactions on Parallel and Distributed Systems, Special Issue on High-Performance Computational Biology, 17(8), pp.773-785, 2006.

[24] Reinhardt, S. and Karypis, G.:" A multi-level parallel implementation of a program for finding frequent patterns in a large sparse graph." In Proceedings of Twenty First IEEE International Parallel & Distributed Processing Symposium, 1-8. Long Beach, Calif.: IEEE Press, 2007.

[25] Ray, A. and Holder, B.:" Efficiency improvements for parallel subgraph miners". In Florida Artificial Intelligence Research Society Conference, FLAIRS '12, 2012.

[26] Belbachir, K., Belbachir, H,: "The Parallelization of Algorithm Based on Partition Principle for Association Rules Discovery", In Proceedings of International Conference on Multimedia Computing and Systems(ICMCS), IEEE, 2012.

[27] Ning, L., Li, Z. , Qing, H. and Zhongzhi, S.:" Parallel Implementation of Apriori Algorithm Based on MapReduce ", International Journal of Networked and Distributed Computing, Vol. 1, No. 2, pp. 89-96, 2013.

[28] Kessl, R., Talukder, N., Anchuri, P., Zaki, M.: "Parallel Graph Mining with GPUs." In: The 3rd International Work-shop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and

© 2017 NSP
Natural Sciences Publishing Cor.

Applications, pp. 1–16, 2014.

[29] Lin, W., Xiao, X. and Ghinita, G.: "Large-scale frequent subgraph mining in MapReduce". In: The IEEE 30th International Conference on Data Engineering (ICDE 2014), pp. 844–855, 2014

[30] Zhao, X., Chen, Y., Xiao, C., Ishikawa, Y., and Tang, J.:" Large Graph Frequent Subgraph Mining Based on Pregel ", Section A: Computer Science Theory, Methods and Tools The Computer Journal, 2016.

[31] Padmapriya, K. M. and Keerthana, M.:" Effective Mapreduces Process for Bigdata Using Load Distribution Fre-quent Sub Graph Mining Algorithm", International Journal of Advanced Research in Computer Science and Soft-ware Engineering , Vol. 6, Issue. 9, pp. 359-366, 2016.

[32] Agrawal, R. and Srikant, R.:" Fast algorithms for mining association rules", in Proc. 20th Int. Conf. Very Large Data Bases, VLDB, edited by J.B. Bocca, M. Jarke, and C. Zaniolo, Morgan Kaufmann 12, pp. 487–499, 1994.

[33] Agrawal, R., Imielinski, T. and wami, A.: "Mining association rules between sets of items in large databases". In: Proc. of the l993ACM on Management of Data, Washington, D.C, pp. 207-216, 1993.

[34] Marghny, H. and Mohammed, D.:"Efficient mining frequent itemsets algorithms", International Journal of Ma-chine Learning and Cybernetics vol.5 Issue. 6, pp.823-833, 2014

[35] Ramraja, T. and Prabhakar, R.:"Frequent Subgraph Mining Algorithms – A Survey ", In Procedia Computer Science, science direct, 47, pp. 197 – 204, 2015.

[36] Bhokare, P. and Sharma, R.:"A Novel Algorithm PDA (Parallel And Distributed Apriori) for Frequent Pattern Mining", In International Journal of Engineering Research & Technology (IJERT), Vol. 3, Issue. 8, ISSN: 2278-0181, 2014.

[37] Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T. and Epema, D.: "A performance analysis of EC2 cloud computing services for scientific computing," in Cloud computing, ser. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, pp. 115–131, 2010.

© 2017 NSP
Natural Sciences Publishing Cor.